다중 시그니쳐에 기반한 변형웜 탐지 기법

이인준*[a], 송치환*[b], 강재우*[c †]
*고려대학교 정보통신대학 컴퓨터전파통신공학과
[a] [e-mail: seminoles@korea.ac.kr]
[b][e-mail: chihwan@korea.ac.kr]
[c][e-mail: kangj@korea.ac.kr]
[†] 교신저자

Multi Signature Based Polymorphic Worm Detection

Injoon Lee*[a], Chihwan Song*[a] and Jaewoo Kang*[c †]
* College of Computer Information & Communication, Korea University, Seoul, South Korea
[a] [e-mail: seminoles@korea.ac.kr]
[b][e-mail: chihwan@korea.ac.kr]
[c][e-mail: kangj@korea.ac.kr]
[†] Corresponding author

요 약

기존의 단일 시그니쳐를 이용한 악성 코드 침입 탐지 시스템은 자신의 컨텐츠를 변형시키는 변형웜을 잡기에는 적합하지 않다. 변형웜을 탐지하기 위한 노력으로 변형웜에 적합한 시그니쳐를 만들기 위한 노력이 있어왔다. 이 연구는 기존의 변형웜 탐지 시그니쳐 방법들을 분석하고 비교하여, 상호 보완적인 멀티 시그니쳐 방법을 제안한다. 이 방법은 정확도 높은 변형웜 탐지 시스템을 구성하기 위한 근본 기술로 활용될 것으로 기대한다.

1. 서론

변형웜은 피해 호스트를 건너 뛸 때 마다 변형 테크닉에 의해 자신의 내용을 변형시키는 악성 코드이다. 이러한 변형웜을 잡기 위한 방법에는 행동 기반 접근법과 내용 기반 접근법이 있다. 내용 기반 접근법은 악성코드 저장소로부터 시그니쳐를 뽑아내어 새롭게 들어오는 악성코드를 잡아내는 방법으로, 이 연구는 내용 기반 접근법에 기초한다.

기존의 침입 탐지 시스템 (IDSs, Intrusion Detection Systems) 은 일반적으로 이미 보유한 악성 코드 시그니쳐들을 새롭게 유입되는 악성 코드와 비교하는 방법을 사용한다. 하지만 네트워크 트래픽의 이상 현상이나 심지어 호스트 PC 가 실질적인 피해를 입고 난 뒤에야 보안 전문가들에 의해 새로운 악성 코드가 파악되고 분석 되어져야 하는 기존 IDSs 의 한계로 변형웜을 포함한 제로 데이 웜의 공격에는 적절한 대응 방법이 될 수 없었다.

시그니쳐 기반의 이러한 IDSs 가 가진 취약점을 보안하기 위해 제로 데이 웜을 탐지하기 위한 연구가 있어왔다. [1]에서는 변형웜들이 자신의 내용을 변형시키더라도, 페이로드에 불변(invariant)하는 컨텐츠들이 있다는 것을 찾아내었다. 변형웜들에 대한 많은 연구들이 이 사실에 근거하여 시그니쳐를 만들어 낸다.

이 연구 역시 변형웜들의 불변 컨텐츠에 집중하여 네트워크로 유입되는 악성 바이너리 파일을 탐지하는 데 목적이 있다. 변형웜 탐지를 위해 가장 널리 사용되는 알고리즘들이 연구되고 구현되고 비교되었으며, 이를 적절히 조합하여 높은 정확도와 효율을 갖는 변형웜 탐지 기법을 제안한다.

이 논문은 다음과 같이 구성된다. 2 장은 기존의 시그니쳐 생성 기술들을 살펴본다. 3 장에서는 변형웜을 탐지하기 위한 시그니쳐들과 시스템을 선보인다. 그리고 4 장에서는 데이터셋을 통한 제안 시스템을 평가하고, 5 장에서 연구를 결론 짓는다.

2. 기존 연구

변형웜 탐지를 위한 내용 기반 접근 방법은 서로 다른 변형웜들에서 공통적으로 나타나는 시퀀스나 패턴을 찾는 것을 목표로 한다. 이 시퀀스와 패턴이 해당 변형웜에 대한 시그니쳐가 되어 새로운 변형웜을 탐지하는데 사용된다.

Snort [2]는 초기 내용 기반 탐지 시스템 중 하나이다. Snort 는 변형웜이 호스트를 감염시키고 난 후에야 웜의 콘텐츠를 새로 유입되는 웜을 탐지하는데 사용한다. 따라서 Snort 의 경량성에도 불구하고, 제로 데이 웜의 탐지는 불가능하다.

Polygraph [1]는 변형웜이 여러 개의 불변 컨텐츠로 구성된다는 사실을 활용하였다. 변형웜을 여러 개의 토큰으로 나누어 일부 토큰들의 연속으로 변형웜을 특징지었다. 토큰들을 사용하는 여러 방법들을 제안하였지만 어떻게 사용하는 것이 더 좋은지에 대한

제안은 하지 못하였다. Polygraph 의 향상 버전으로 Hamsa [3]와 LISABETH [4]가 제안되었다. 역시 멀티 토큰을 사용 하였고, Polygraph 의 취약성과 처리 속도를 향상시키는 데에 집중하였다.

Autograph [5], 는 Rabin fingerprints 를 사용하여 트래픽을 여러 개의 작은 블록 들로 나눈다. 블록 들로부터 가장 자주 등장하는 바이트로 변형웜의 시그니쳐를 만들어낸다. 반면 Honeycomb [6]는 의심군으로부터 가장 긴 부분시퀀스를 찾아내어 시그니쳐로 사용한다. EarlyBird [7]는 모든 컨텐츠가 해쉬 함수에 의해 기록되어 변형웜 탐지를 위해 사용된다. 세 가지 시스템의 공통된 문제점은 단일 연속 부분시퀀스를 사용하기 때문에, 변형웜에서 불변 컨텐츠가 아주 작을 경우 효율적으로 잡을 수 없다는 단점이 있다.

3. 다중 시그니쳐의 생성

3.1. 기본 알고리즘

Longest Common Subsequence (LCSeq): LCS 는 본래 분자시퀀스 분석에 있어서 동일한 가장 긴 서브시퀀스를 찾기 위해 개발되었다[10]. 이 알고리즘은 시퀀스 매칭을 필요로 하는 많은 분야에 적용되었고, [1]과 [6]은 이 LCS 알고리즘에 기초하여 변형웜을 잡아낸다.

LCS 의 기본 알고리즘은 다음과 같다. 두 시퀀스 n, m 이 접두사 $n_1n_2\cdots n_i$ 와 $m_1m_2\cdots m_j$ 을 갖는다고 할 때, 두 시퀀스의 LCS 스코어 $s_{i,j}$ 는 아래의 식을 만족한다:

$$s_{i,j} = \max\begin{cases} s_{i-1,j} \\ s_{i,j-1} \\ s_{i-1,j-1}+1, & \text{if } n_i = m_j \end{cases} \quad (1)$$

i 와 j 가 0 일 때, LCS 스코어는 0 이다. 다이나믹 프로그래밍을 사용하여 LCS 스코어의 최대값과 스코어에 대한 서브시퀀스를 구할 수 있다. 그림 1 은 'oxnxexzxtwoxxw' 과 'ytwoyownyeyz' 문자열의 LCS 결과이다.

	y	t	w	o	y	o	w	n	y	e	y	z
o	0	0	0	1	1	1	1	1	1	1	1	1
x	0	0	0	1	1	1	1	1	1	1	1	1
n	0	0	0	1	1	1	1	2	2	2	2	2
x	0	0	0	1	1	1	1	2	2	2	2	2
e	0	0	0	1	1	1	1	2	2	3	3	3
x	0	0	0	1	1	1	1	2	2	3	3	3
z	0	0	0	1	1	1	1	2	2	3	3	4
x	0	0	0	1	1	1	1	2	2	3	3	4
t	0	1	1	1	1	1	1	2	2	3	3	4
w	0	1	2	2	2	2	2	2	2	3	3	4
o	0	1	2	3	3	3	3	3	3	3	3	4
x	0	1	2	3	3	3	3	3	3	3	3	4
x	0	1	2	3	3	3	3	3	3	3	3	4
w	0	1	2	3	3	3	4	4	4	4	4	4

LCS score = 4

LCS signature: onez

(그림 1) 문자열 'oxnxexzxtwoxxw' 과 'ytwoyownyeyz' 의 LCS 예제: LCS 스코어 4 와 서브시퀀스 'onez' 가 출력되었다.

Simplified Regular Express (SRE) [11]: LCS 의 확장 버젼인 SRE 는 시퀀스 매칭 문제를 푸는 데에 한 가지 큰 장점을 갖는다. SRE 는 정규 표현식의 개념을 가져와서 공통 서브시퀀스 사이에 존재하는 공백까지도 특징지을 수 있다. 예를 들어, 변형웜 중 하나인 Code Red II 의 경우 불변 컨텐츠 '%u' 와 '%u7801' 사이에 항상 4 바이트의 랜덤 문자를 가진다. LCS 를 사용할 경우 '%u%u7801' 만을 시그니쳐로 잡아내지만, SRE 의 경우 '%u[4]%u7801' 와 같이 4 바이트의 공간을 시그니쳐에 포함시킨다. 추가적으로 '*' 은 0 부터 무한 바이트의 문자와 '?' 은 1 부터 무한 바이트의 문자를 나타낸다.

SRE 를 구현함에 있어서 연속적으로 매칭되는 바이트에 대해서 추가적인 점수를 부여했다. 연속적인 매칭 바이트에 3 점을 추가함으로써, 문자의 연속과 비연속에 더 집중하는 SRE 특징을 강조할 수 있다.

그림 2 는 앞의 LCS 예제에서 사용한 동일한 문자열 'oxnxexzxtwoxxw' 과 'ytwoyownyeyz' 문자열의 SRE 알고리즘 결과이다.

	y	t	w	o	y	o	w	n	y	e	y	z
o	0	0	0	1	1	1	1	1	1	1	1	1
x	0	0	0	1	1	1	1	1	1	1	1	1
n	0	0	0	1	1	1	1	2	2	2	2	2
x	0	0	0	1	1	1	1	2	2	2	2	2
e	0	0	0	1	1	1	1	2	2	3	3	3
x	0	0	0	1	1	1	1	2	2	3	3	3
z	0	0	0	1	1	1	1	2	2	3	3	4
x	0	0	0	1	1	1	1	2	2	3	3	4
t	0	1	1	1	1	1	1	2	2	3	3	4
w	0	1	5	5	5	5	5	5	5	5	5	5
o	0	1	5	9	9	9	9	9	9	9	9	9
x	0	1	5	9	9	9	9	9	9	9	9	9
x	0	1	5	9	9	9	9	9	9	9	9	9
w	0	1	5	9	9	9	10	10	10	10	10	10

SRE score = 10

SRE signature: ?two??w

(그림 2) 문자열 'oxnxexzxtwoxxw' 과 'ytwoyownyeyz' 의 SRE 예제: SRE 스코어 10 과 서브시퀀스 '?two??w' 가 출력되었다.

일반적으로 LCS 와 SRE 점수는 비교 시퀀스의 길이가 길수록 커지기 때문에 다음의 정규화를 하였다.

$$\text{Score}_{(i,j)} = \frac{S_{i,j}}{\min(|i|,|j|)} \quad (3)$$

만약, $|i|$나 $|j|$가 너무 작은 경우, 스코어가 역시 증가하기 때문에, 최소 길이를 제한하였다.

Naïve Bayesian [12]: 베이지안 분류기는 Bayesian 통계적 분석에 기반한 확률 분류기이다. 일반적으로 베이지안 분류는 큰 데이터 베이스에서도 높은 정확도와 빠른 속도를 보여 준다. 나이브 베이지안은 각 어트리뷰트들이 독립적인 확률 분포를 갖는다는 가정을 한다. 주어진 데이터 셋에서 P($malware|X$)는 X 가 $malware$ 에 속할 확률이다. 만약 P($malware$|X)가 P($\neg malware|X$)보다 크다면, X 는 $malware$ 로 분류된다. 베이지안 이론에 따르면,

$$P(malware|X) = \frac{P(X|malware)P(malware)}{P(X)},$$

$$P(\neg malware|X) = \frac{P(X|\neg malware)P(\neg malware)}{P(X)}$$ 를 만족한다. P(X)는 모든 클래스에서 공통이기 때문에, P($malware|X$)P($malware$)와 P($\neg malware|X$)P($\neg malware$)만 계산하면 된다. P($malware$) 와 P($\neg malware$)는 각 클래스에 속한 멤버 개수를 세면 되고, P($malware|X$)와 P($\neg malware|X$)는 나이브 베이지안 가정에 의해,

$$P(malware|X) = \prod_{k=1}^{n} P(x_k|malware),$$

$$P(\neg malware|X) = \prod_{k=1}^{n} P(x_k|\neg malware)$$ 가 된다.

File extension and file size: [13] 논문에 의하면, 모든 악성코드는 실행 가능한 코드를 포함한다. 이 연구는 네트워크를 통해 유입되는 바이너리 파일 형태의 변형웜을 고려하기 때문에, 파일 확장자를 기록하여 변형웜의 특징 중 하나로 사용하였다.

바이너리 파일의 파일 크기 역시 저장되어 변형웜을 분류하는데 사용되었다. 이런 특징은 변형웜을 탐지하는 데에 다소 모호하지만, 실험결과를 통해 약간의 정확도 향상을 보였다.

3.2. 멀티 시그니쳐

LCS/SRE 시그니쳐: LCS/SRE 알고리즘을 사용하여 LCS/SRE 시그니쳐를 만든다. LCS/SRE 방법이 두 시퀀스간의 유사도에 기반하기 때문에 변형웜 저장소의 악성코드들만 사용하여 다음의 순서로 시그니쳐를 만들어낸다.

1. 저장소의 모든 악성코드 페어에 대한 LCS/SRE 스코어를 구한다. 만약 스코어가 정해진 쓰레스홀드를 넘으면 두 악성코드는 하나의 클러스터에 속하게 된다. 모든 페어에 대한 LCS/SRE 스코어 계산이 끝나면 각 악성코드는 0개부터 여러 개의 클러스터에 속하게 된다.
2. 하나의 클러스터에 속한 멤버들에 대하여 가장 유사도가 가까운 페어부터 시작하여 모든 멤버에 대한 LCS/SRE 서브시퀀스를 구한다.
3. 1-2 스텝은 LCS와 SRE에 다르게 적용된다. 2 스텝 후 LCS와 SRE의 시그니쳐를 클러스터의 개수만큼 구하게 된다.

베이지안 시그니쳐와 스코어: 베이지안 계산에 사용되는 어트리뷰트를 정의하기 위하여, 모든 악성코드 저장소와 정상 바이너리 저장소에서 각 2 번 이상 나타나고 길이 10 이상의 유일한 서브 시퀀스를 어트리뷰트라 정의하였다. 각 어트리뷰트는 전체 저장소의 악성코드와 정상 바이너리에서 몇 번 등장하는지 기록된다. 결과적으로 이 과정은 베이지안 계산을 위한 각 어트리뷰트들의 $P(x_k|malware)$와 $P(x_k|normal)$를 구하게 된다. 여기서 $P(x_k|malware)$값과 $P(x_k|normal)$를 베이지안 시그니쳐라 정의하였다.

일반적으로 베이지안 분류법에서 $P(malware|X)$와 $P(normal|X)$을 비교하여 큰 값의 클래스를 선택하게 되지만, 실험 결과 악성코드의 경우에도 $P(normal|X)$가 상대적으로 더 큰 값을 보여주기 때문에 베이지안 스코어로 $log(P(malware|X))/log(P(\neg malware|X))$를 사용하였다.

3.3. C4.5 결정 트리를 이용한 다중 시그니쳐

이 연구에서 다중 시그니쳐를 사용하기 위한 방법으로 C4.5 결정 트리를 사용하였다.

6-tuple 어트리뷰트: 바이너리 파일 하나 당 앞서 소개한 3 가지 스코어와 파일 확장자, 크기, 클래스(악성코드/정상)의 6 가지 항목으로 그림 3 과 같은 C4.5 결정 트리의 어트리뷰트를 정의한다.

LCS_score	SRE_score	Baye_score	extension	size	class

(그림 3) 바이너리 파일에 대한 6-tuple 어트리뷰트: LCS/SRE/베이지안 스코어와 파일 확장자, 파일 크기, 클래스(악성코드/정상)으로 구성된다.

4. 평가와 결과

4.1. 데이터셋

[1, 6, 9, 16, 17 와 18]를 통해 다양한 변형웜 종류를 파악하였다. [1, 4 와 19]와는 다르게 실험을 위해 인위적으로 만들어진 변형웜이 아닌 실제 변형웜 샘플을 사용하였다. 변형웜 샘플들은 'www.offensivecomputing.net' 에서 앞서 파악된 변형웜 이름을 쿼리로 입력하여 구할 수 있었다. 다양한 anti-virus 엔진으로부터 일관성을 유지하기 위하여 BitDefender 가 분류한 관련 변형웜 만을 취하였다.

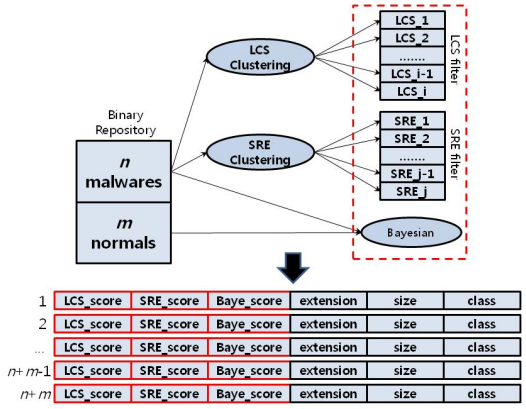
결과적으로, 총 125 개의 변형웜 샘플을 확보하였고, 18 개의 *Chameleon*, 6 개의 *Code Red*, 27 개의 *Ply*, 17 개의 *Tequila*, 18 개의 *V2PX* 그리고 40 개의 *Vienna* 로 구성된다.

비교 대상인 정상 바이너리는 Windows Vista OS 의 'windows/system32' 폴더에서 임의로 추출한 250 개의 바이너리 파일이 사용되었다. 변형웜과 정상 바이너리 샘플은 '*http://infos.korea.ac.kr/malsig/binaries.zip*' 에서 다운로드 받을 수 있다.

4.2. 평가 방법

데이터 셋이 크지 않기 때문에 정확도 평가를 위하여 leave-one-out cross-validation 방법을 사용하였다. 총 프로세스는 다음의 8 단계를 거친다:

1. 저장소에서 (375개의 바이너리 파일) 1개의 파일을 테스트 셋으로 추출한다.
2. 저장소에서 (374개의 바이너리 파일) 1개의 파일을 결정 트리용 트레이닝 셋으로 추출한다.
3. 남은 373개의 바이너리 파일을 이용하여 LCS/SRE/베이지안 시그니쳐를 만든다.
4. 2번 스텝의 트레이닝 셋으로 LCS/SRE/베이지안 스코어를 구한다.
5. 트레이닝 셋의 스코어들과 파일 확장자, 크기, 클래스 정보를 합하여 6-tuple 어트리뷰트를 만든다. 그림 4는 1-5 스텝의 C4.5 트리 용 어트리뷰트를 만들기 위한 전체 흐름을 보여준다.
6. 2번부터 5번 스텝을 374개의 파일에 반복한다. 결과적으로 374개의 6-tuple 어트리뷰트 셋을 얻고, 어트리뷰트 셋을 통해 C4.5 결정 트리를 만든다.
7. 1번 스텝에서 추출한 테스트 셋을 C4.5 결정 트리에 넣어 결과를 얻는다.
8. 1번부터 7번 스텝을 375번 반복하여 C4.5 결정 트리의 정확도를 얻는다.



(그림 4) 결정 트리를 만들기 위한 전체 흐름도: 저장소의 악성 코드와 정상 바이너리를 이용하여 6-tuple 어트리뷰트를 완성한다.

4.3. 결과

제안된 시스템은 C4.5 결정 트리의 각 어트리뷰트를 제한하여 얻은 베이스 라인 방법과 비교되었다. 표 1 은 우리의 시스템과 베이스 라인들과의 정확도를 비교한 표이다. *malware* 로 분류된 것을 positive 로 보았고 반대의 경우 negative 로 보았다.

F-score 만을 고려한 경우, 베이지안만 사용한 베이스 라인이 가장 높은 스코어를 기록하였고, 제안된 시스템이 2.3% 더 낮게 나왔다. 하지만 recall 을 고려하였을 때, 제안된 시스템 만이 100%를 기록하였고, 이것은 모든 변형웜을 탐지하였다는 것을 의미한다. IDSs 에서 false negative 가 더 치명적이기 때문에, 제안된 시스템이 가장 좋은 성능을 보인 것을 알 수 있다.

<표 1> 성능 비교 테이블: 제안한 시스템(맨 아래)와 다른 베이스라인들과의 정확도를 비교하였다.

	precision	recall	F-score
LCS only	44.4%	94.4%	60.4%
SRE only	50.0%	59.2%	54.2%
Bayes only	100.0%	98.4%	99.2%
LCS & SRE & Bayes	99.2%	98.4%	98.8%
LCS & SRE & Bayes & Extension	93.3%	100.0%	96.5%
LCS & SRE & Bayes & SIZE	99.2%	98.4%	98.8%
LCS & SRE & Bayes & Extension & SIZE	94.0%	100.0%	96.9%

표 1 에 대한 더 자세한 정보는 '*http://infos.korea.ac.kr/malsig/Data_and_Result.pdf*' 에서 다운로드 받을 수 있다.

많은 변형웜 탐지 시스템들이 LCS, SRE 또는 베이지안 알고리즘에 기초하고 있지만, 이 연구의 결과는 이들 중 하나만 사용했을 때는 변형웜의 모든 변형을 탐지하기에 충분하지 않다는 것을 보인다.

5. 결론

이 연구에서, 우리는 변형웜을 탐지하기 위해 널리 사용되는 3 가지 알고리즘을 검증하고 활용하였다. 각 알고리즘의 장점을 최대화하고, 약점을 보완하여, 제안된 시스템은 실제 변형웜 데이터를 대상으로 하여 높은 정확도와 재현율을 보였다.

이 시스템은 네트워크에 설치 되었을 때, 실시간 업데이트를 고려에 두고 설계 되었지만, 자세한 방법과 계산량을 줄이는 방법에 대하여는 이후의 논문에 소개하도록 하겠다.

참고문헌

[1] J. Newsome, B. Karp and D. Song, "Polygraph: Automatically Generating Signatures for Polymorphic Worms," *IEEE Symposium on Security and Privacy*, 2005.

[2] Martin Roesch, "Snort – Lightweight Intrusion Detection for Networks," In *Proceedings of LISA, 13th Systems Administration Conference*, 1999.

[3] Zhichun Li, Manan Sanghi, Yan Chen, Ming-Yang Kao and Brian Chavez, "Hamsa: Fast Signature Generation for Zero-day Polymorphic Worms with Provable Attack Resilience," In *Proceedings of the IEEE Symposium on Security and Privacy*, 2006.

[4] Lorenzo Cavallaro, Andrea Lanzi, Luca Mayer and Mattia Monga, "LISABETH: Automated Content-Based Signature Generator for Zero-day Polymorphic Worms," *Software Engineering for Secure Systems*, 2008.

[5] Hyang-Ah Kim and Brad Karp, "Autograph: toward automated, distributed worm signature detection," In *Proceedings of the 13th conference on USENIX Security Symposium*, 2004.

[6] C. Kreibich and J. Crowcroft, "Honeycomb – Creating Intrusion Detection Signatures Using Honeypots", *ACM SIGCOMM Computer Communications Review*, 2004.

[7] Sumeet Singh, Cristian Estan, George Varghese and Stefan Savage, "Automated Worm Fingerprinting," In *Proceeding of the 6th conference on Symposium on Operating Systems Design & Implementation*, 2004.

[8] T. F. Smith and M. S. Waterman, "Identification of Common Molecular Subsequences," *J. Mol. Biol.*, 1981.

[9] Y. Tang, X. Lu, B. Xiao, "Generating Simplified Regular Expression Signatures for Polymorphic Worms", *Springer-Verlag Berlin Heidelberg*, 2007.

[10] J. Han, M. Kamber, "Data Mining : Concepts and Techniques", *Morgan Kaufmann Publishers*, p.311-315, Second Edition, 2006.

[11] Christopher Kruegel, Engin Kirda, Darren Mutz, William Robertson and Giovanni Vigna, "Polymorphic Worm Detection Using Structural Information of Executables," *Recent Advances in Intrusion Detection*, 2006.

[12] Ke Wang, Gabriela Cretu and Salvatore J. Stolfo, "Anomalous Payload-Based Worm Detection and Signature Generation," *Recent Advances in Intrusion Detection*, 2006.

[13] Kaspersky Lab viruslist, *'http://www.viruslist.com'*.

[14] Symantec Security Article, '*http://www.securityfocus.com/infocus/1671*'

[15] G DATA Security Labs, History of Malware, '*http://sa.gdatasoftware.com/security-labs/information/history-of-malware/1988-1994.html*'

[16] Ki Hun Lee, Yuna Kim, Sung Je Hong and Jong Kim, "PolyI-D: Polymorphic Worm Detection Based on Instruction Distribution," *Information Security Applications*, 2007.